DOCUMENT RESUME

ED 148 303 IR 005 255

AUTHOR Hultgren, Jan; Larsson, Rolf
TITLE A Method Defining a Limited Set of Character-Strings with Maximal Coverage of a Sample of Text.
INSTITUTION Royal Inst. of Tech., Stockholm (Sweden). Library.
REPORT NO TRITA-LIB-4020a
PUB DATE Jun 73
NOTE 11p.

EDRS PRICE MF-$0.83 HC-$1.67 Plus Postage.
DESCRIPTORS Computer Programs; Data Bases; *Electronic Data Processing; *Information Processing; *Information Retrieval; *Information Storage; *Programing Languages

ABSTRACT

This is a progress report on a project attempting to design a system of compacting text for storage appropriate to disc oriented demand searching. After noting a number of previously designed methods of compression, it offers a tentative solution which couples a dictionary of most frequent character-strings with a set of variable-length codes. The dictionary is not restricted to natural language words or to content-bearing strings. A suite of IBM 360 assembly language programs has been constructed to count frequencies and to compute a set of parameters which will mitigate constructing a dictionary. A program to sort and print the file resulting from the other programs has also been developed. (WBC)

****
* Documents acquired by ERIC include many informal unpublished *
* materials not available from other sources. ERIC makes every effort *
* to obtain the best copy available. Nevertheless, items of marginal *
* reproducibility are often encountered and this affects the quality *
* of the microfiche and hardcopy reproductions ERIC makes available *
* via the ERIC Document Reproduction Service (EDRS). EDRS is not *
* responsible for the quality of the original document. Reproductions *
* supplied by EDRS are the best that can be made from the original. *
****

REPORT TRITA-LIB-4020a JUNE 1973

ED148303

U.S. DEPARTMENT OF HEALTH, EDUCATION & WELFARE
NATIONAL INSTITUTE OF EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

A METHOD DEFINING A LIMITED SET OF CHARACTER-STRINGS WITH MAXIMAL COVERAGE OF A SAMPLE OF TEXT

Jan Hultgren and Rolf Larsson

Royal Institute of Technology Library
S-100 44 STOCKHOLM 70

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

Stephan Schwarz

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC) AND USERS OF THE ERIC SYSTEM."

IR005255

A METHOD DEFINING A LIMITED SET OF CHARACTER-STRINGS WITH MAXIMAL COVERAGE OF A SAMPLE OF TEXT.

Several bibliographic data-bases, among which are those from Chemical Abstracts, ISI and INSPEC, are being searched within the fairly extensive current awareness service of the library of the Royal Institute of Technology in Stockholm. When considering the transition from the normal serial search mode of most SDI systems to a disc oriented demand search system (be it terminal-oriented or not), the glaring inefficiency of the conventional coding methods when applied to textual data, their abundance in bibliographic files, and the heavy cost of storage force one to the conclusion that one of the primary targets for investigation should be methods for compacting text. To investigate the possibilities and to design a system of compression has been one of the aims of a SINFDOK supported project at the library. This is to report its progress.

A variety of methods of compression have recently been discussed in the literature. Thus Heaps[1], with the help of a huge dictionary of natural language words and a set of variable-length codes, has reached reductions of up to 80%. Hunt[2] and others[3], by using fixed-length codes for the most frequent pairs of alphabetic characters (digrams), have compressed their files to 60% of the original size. As the length of their code is equal to that of one character, they can, however, never obtain a better figure than 50%. Lynch[4] has considered a form of run-length coding involving; first a conversion of the normal character ordigram representation on a set of codes with the property that the higher the frequency of the character is the higher is the ratio of zero-bits to one-bits of its code, secondly a compressive coding by counting the length of the runs of zero-bits in the converted representation. The rate of compression obtained has, however, not been greater than that of the digram method.

Our, as yet tentative, solution to the problem of compression is to couple a dictionary of the most frequent character-strings, be they content-bearing or not, with a set of variable-length codes. The number of strings thus chosen should be small enough for the entire dictionary economically to be kept in primary storage while coded text is being decoded. The size range aimed at for decoder and dictionary is 50 000 - 150 000 bits wherein 1000 - 3 000 different character strings would be recognized. These different strings would be assigned codes from a set of codes of two, three or four different lengths of which the longest, as it would have to account for the majority of strings, would occupy 12 - 16 bits. To consider only natural language words for the dictionary we find an unnecessary restriction as the morphological structure of most natural languages (albeit less with English than with some others) makes it highly likely that among the most frequent strings would be found many shorter than words but yet content-bearing (prefixes, suffixes, stems, roots, inflexions, etc.) Again the word may well be too short a unit as certain syntactic features are to be expected to occur frequently (prepositional adjuncts, the of-genitive in English, etc.) On the other hand we do not want to exclude the advantages of the digram-method: that many very frequent combinations of characters are non-content-bearing (th, st, tr, etc); rather would we, as Lynch suggests[5], extend its use to n-grams with n greater than 2.

However, our proposed method poses some problems not encountered by Heaps, nor by Hunt and his followers, both in regard to the counting of frequences and the subsequent design of a dictionary and in regard to the actual coding of a text. For while Heaps has given himself a natural unit for both counting and coding purposes by considering words only and while Hunt, thanks to restricting himself to digrams, finds counting easy, we must somehow design methods: for counting the frequencies in a material of every string of every length up to some undecided but rather great number, for deciding on which strings shall make up our dictionary, and for coding a text with the help of a dictionary, which coding by necessity must contain some sort of look-ahead-function to ensure an efficient use of the dictionary. Of course the same sort of problem with the coding is met by Hunt, but its magnitude increases rapidly with the maximum length of the strings in the dictionary. We have not yet decided in detail how to handle the coding, but have composed a suite of IBM 360 assembly language programs to take care of the

counting of frequencies and to compute a set of parameters which will mitigate the task of constructing a dictionary.

The bibliographic data-bases being studied consist of references, each of which is made up of categories such as: journal, author, title, key-words, etc. The first of our programs is governed by parameters set by the user: the categories to be processed; the total number of characters to be processed, n; the length of the strings computed by the program, an integral number m, not greater than 16. Within a reference the program moves a window of length m, from left to right and character by character, through a wanted category, which has first been padded to the right with m-1 characters non-occurent in the data-base. Each move produces an out-record consisting of a symbol for the category and the string within the window. When a reference is processed, the total number of out-records so far produced is compared with n and if it exceeds this set number the processing is stopped. The out-records are then sorted alphabetically within categories.

The second program is designed to compute the number of occurences, which we shall call the absolute frequency, of each distinct string of each of the lengths 1,2,...,m, and to produce a table containing the distribution of strings after length of string and after interval of absolute frequency, where the intervals are given by the user (for an example see table 1). Furthermore the program computes the weighted frequencies (i.e. the product of length and absolute frequency) and produces a file containing the strings (to a number set approximately by the user) with the highest weighted frequencies and giving length, absolute, and weighted frequency of the string. A string will not be considered for the file if there is in the material a longer string beginning with the shorter string and having the same absolute frequency. The criterion that the longer string shall actually begin with the shorter is of course too strong; we are not interested in any substring with the same absolute frequency, but the deletion of the other substrings will for technical reasons have to be deferred to the next step.

While testing the programs, we found from printing the file just described that there were obviously strings, longer than m characters, frequent enough to qualify for the file. Although it was impossible with absolute certainty

to construct the longer strings present in the original material from this file, we considered the detrimental effects of misconstruction negligible and designed a program to construct these longer strings. This program compares strings of the same absolute frequency and the same length, m, with each other and if two strings both contain the same substring of length, m-1, the one is used to extend the other. The extension is not carried further than to 380 characters, in spite of the fact that the original text contained a number of strings longer than 396 characters. These, however, seem to be accounted for by the singularity of the studied material, and we believe that if studies are carried out on a sampled material from different points of time and from different data-bases the longer strings will be of little or no importance.

The strings used to construct larger strings are exchanged for the extended strings and the amended file is sorted in descending absolute frequency order and then in descending length order. The sorted file is then processed by a program which deletes every string's substrings if they have the same absolute frequency. Now, it is obviously one of the basic ideas of this approach to compression that it may well be worthwhile to have in the dictionary not only a given string of high frequency, but also one or more of its substrings; it is equaly obvious that if the substring's frequency differs only little from that of the string itself, then the substring should not be considered for the dictionary. Therefore the program computes a reduced absolute frequency for each string in the following manner. Reduction does not take place for strings which are not substrings of some other string. If a string is also a substring, its absolute frequency is reduced once and once only and then with the highest of the absolute frequencies of the strings whereof itself is a substring. The efficiency of compression depends of course on the weighted frequency, but it also depends on the code-length. The user of the program is allowed to specify a code-length (in number of bits, the system is implemented on an 8-bit byte oriented computer) as a parameter and the program produces a file of records consisting of: string, length, absolute frequency, reduced absolute frequency, weighted frequency, weighted frequency computed from length reduced by code-length, weighted frequency computed from reduced absolute frequency, and weighted frequency computed from reduced length and reduced absolute frequency. It is true that we intend to use variable-length codes, but the

number of codes shorter than the maximum length will be relatively small and we hope that the likely candidates for these codes will be rather obvious in the material, so that the restriction to just one code-length in the program will not prove serious.

The last program,so far, is just a sorting and printing routine applied to the resulting file from the other programs, so that a desired number of the strings sorted after a desired combination of criteria can be displayed. Some examples are given in table 2.

The described set of programs we judge to be a necessary tool for our approach to text compaction, which can be epitomized as follows:

1. Let N be the number of bits available for a decoder kept in primary storage
2. Let $S = (s_0,s_1,\ldots,s_{n-1})$ be a set of variable-length strings such that it can be kept within a decoder of N bits and let $C = (c_0,c_1,\ldots,c_{n-1})$ be a set of codes
3. Order the strings $s_0,s_1,\ldots,s_{n-1}$ in decreasing order of their coverage of a suitable sample of textual data
4. Assign code c_i to string s_i; $i = 0,1,\ldots,n-1$
5. Devise a coding algorithm which by partitioning text in some suitable series of strings of S minimizes the sum of the lengths of the corresponding series of codes

So far efforts have been spent mainly on creating a tool for the better selection of candidate strings for assignment to a fairly small number of codes. The final selection of strings must in all likelihood be postponed till the coding algorithm with its look-ahead-function is ready; the items 1 through 5 are obviously not independent, and it would be fatuous indeed to claim that, with the help of our method, it will be possible to give an optimal solution to the compaction problem. However, in comparison with, for example the digram method considerable gains are expected. But before a coding algorithm is considered in detail the obvious step to take is to apply the tool described here to samples from different data-bases from different points of time and to compare the different candidate lists in order at least to identify the common subset of candidate strings.

290031 RECORDS PROCESSED; TYPE=9: 132382 RECORDS PROCESSED; MASKLENGTH=16;

STRING LENGTH THRESHOLD	1	2	3	4	5	6	7	8
2	1	206	1930	7435	16448	26716	37113	47089
3	3	102	736	2337	4269	5733	6527	6814
4		63	396	1325	2127	2416	2509	2393
5		45	288	838	1219	1329	1341	1192
6		36	219	547	806	802	766	651
7		37	167	483	557	580	534	465
8	1	16	122	321	388	413	397	332
9		27	108	295	344	343	308	245
10		13	90	240	292	271	221	186
11	1	15	87	186	212	212	184	134
12		5	50	157	173	158	135	105
13		4	64	155	162	154	142	106
14		6	61	143	150	120	81	58
15		9	50	131	130	105	73	57
16		11	50	110	109	87	84	87
17		10	42	103	90	71	68	57
18		8	59	82	80	71	55	46
19		2	33	87	75	59	48	41
20	1	4	36	62	60	50	34	25
21		4	30	47	47	42	35	29
22		9	29	58	48	41	36	30
23	1	3	24	57	39	39	28	25
24		6	35	39	52	41	30	19
25		4	46	49	41	24	18	12
	44	356	1071	1038	825	664	520	415
SUM	52	1001	5823	16325	28743	40541	51287	60613

Table 1, page 1. 132 K title characters from the INSPEC database.

WRITELIMIFACTOR= 2;NO. OF RECORDS ON CHOPSTEW= 15484

9	10	11	12	13	14	15	16	SUM
55580	62721	68650	73241	76697	79114	80766	81783	715490
6639	6372	5841	5266	4651	4090	3588	3160	66128
2147	1884	1630	1344	1093	889	718	578	21512
1028	866	715	582	474	369	300	243	10829
562	452	336	251	186	144	120	89	5967
361	278	206	151	118	102	83	72	4194
270	220	175	153	134	114	86	67	3209
196	147	117	94	80	65	52	40	2461
143	107	79	53	37	29	24	17	1802
117	94	70	47	30	22	14	10	1435
79	55	41	29	19	15	11	7	1039
89	74	68	57	51	40	33	30	1229
52	42	28	18	10	7	4	3	783
39	26	18	12	6	5	4	2	667
95	92	91	88	89	92	95	100	1280
41	29	24	17	11	7	6	5	581
39	32	25	22	19	18	16	14	586
32	27	21	16	14	11	8	5	479
18	16	9	11	12	11	8	5	362
24	16	13	8	6	4	3	3	311
30	19	8	3					311
13	8	6	3	3	2			251
12	8	4	5	7	4	3	2	257
6	4	3	3		1	1	1	213
351	303	267	244	225	210	203	198	6934
67963	73892	78445	81718	83972	85365	86146	86434	848320

Table 1, page 2.

ORDI-NAL	LEN.	ABS-FREQ	ABS-RED	WFREQ	WFREDAB	WFREDL	WFREDABL	STRING
50	26	57	57	1482	1482	1396	1396	ASSOCIATION_OF_
51	2	1043	691	2086	1382	521	345	RO
52	19	72	72	1368	1368	1260	1260	_(ABSTRACTS_ONL
53	2	1138	682	2276	1364	569	341	OR
54	2	839	671	1678	1342	419	335	SI
55	5	920	266	4600	1330	3220	931	TION_
56	2	1098	651	2196	1302	549	325	EN
57	2	1091	651	2182	1302	545	325	TR
58	2	1724	646	3448	1292	862	323	_T
59	2	923	642	1846	1284	461	321	Y_
60	22	58	58	1276	1276	1189	1189	_EXPLORATION_GE
61	5	336	249	1680	1245	1176	871	S_OF_
62	2	1068	617	2136	1234	534	308	CT
63	2	1054	609	2108	1218	527	304	R_
64	3	569	398	1707	1194	853	597	S__
65	1	2177	1165	2177	1165			G
66	1	2658	1157	2658	1157			H
67	16	72	72	1152	1152	1044	1044	S_42ND_NATIONAL_
68	2	1269	575	2538	1150	634	287	D_
69	3	782	383	2346	1149	1173	574	_IN
70	2	1032	574	2064	1148	516	287	_M
71	2	1017	559	2034	1118	508	279	ME
72	2	739	559	1478	1118	369	279	IT
73	2	1294	558	2588	1116	647	279	AL
74	2	1130	556	2260	1112	565	278	_C

Table 2 (based on 132 K title characters from the INSPEC database).

References

1 Heaps, H. S., Storage Analysis of a Compression Coding for Document Data Bases, INFOR, vol.10, pp 47 - 61 (1972).

2 Snyderman, M. and Hunt, B., The Myriad Virtues of Text Compaction, Datamation, pp 36 - 40, (Dec. 1, 1970).

3 Scheiber, W. D. and Thomas, G. W., An Algorithm for Compaction of Alphanumeric Data, J. Library Automation, 4, pp 198 - 206, (1971).

4 Lynch, Michael F., Compression of Bibliographic Files Using an Adaptation of Run-length Coding, Inform.Stor. Retr., vol. 9, pp 207 - 214, (1973).

5 Lynch, Michael F., Studies in Data-base Characterisation and Organisation, Paper presented at the OSTI Workshop on String Processing Techniques, (Nov. 20 - 21, 1972).